SPECIFICATION AMENDMENTS

Please amend the specification as follows:

Substitute paragraph [0003] at page 2 with the following:

As a result, sensitive information such as financial data, account numbers, identification indicia and the like need to be transmissible while preserving confidentiality via such networked media. Accordingly, there ~~are needs~~ is a need to verify user identity and to preserve secrecy with respect to information and with respect to granting access to information while still permitting appropriate access to such resources and facilities.

Substitute paragraph [0008] at page 4 with the following:

A problem with this scheme is that the system can be attacked by sending numerous sham names to the system, possibly even by spreading a virus that causes recipient computers to transmit such sham requests. The process of looking up keys typically involves delay in order to search the database and this results in a "bottleneck" which cripples the overall authentication system when numerous sham names are used ~~result in need~~ to cause the system to attempt to look up non-existent keys. As a result, even limited resources may be able to bottleneck the system such that it may be unable to respond to legitimate clients presenting legitimate credentials in order to try to gain access to the system. Such is known as a "denial of service" attack.

Substitute paragraph [0009] at page 5 with the following:

Accordingly, there are needs for improved techniques, systems and ~~apparatus~~ apparatuses for providing both more robust security (e.g., involving a larger amount of authentication data) together with rapidly and efficiently avoiding becoming too encumbered by bogus or sham requests without denying legitimate clients timely access to ~~such~~ available facilities.

Substitute paragraph [0029] at page 11 with the following:

One way to encrypt and also reduce the amount of data that is stored is to generate a "hash" or digest corresponding to the user name, also referred to herein as a NameHash. A hash or digest of data is a number that can be repeatedly computed from the data and that can be truncated to require less memory to store because the truncated hash digest is shorter than the data from which it is derived. Many hash algorithms result in a hash digest that does not provide enough information to permit the data from which the hash digest was derived to be computed and is not easily faked. Examples of well-known hash algorithms include the ~~Hashed Message Authentication Code As~~ Message Digest 5 ([[HMAC-]] MD5: 2 times 128 bit hashing) and the ~~Hashed Message Authentication Code Message~~ Secure Hash Algorithm ([[HMAC-]] SHA: 2 times 160 bit hashing). Additionally, data in the cache memories 155 and/or 160 may be organized into hash tables, which facilitate rapid data accession (e.g., name and key pair) using the hash digest of the name as a ~~cachekey~~ cache key. However, other forms of data compression coupled with encryption may be employed and are to be understood to be within the scope of the term "hash" as used herein.

Substitute paragraph [0050] with the following:

In a block 315, first data corresponding to each of the new/revised client records are entered into the primary cache memory 155 of Fig. 1. In one embodiment, the first data include the client name and/or the client key or data derived from such. In one embodiment, the first data include the NameHash and/or the NameKeyHash. In one embodiment, the NameHash is used as a ~~cachekey~~ cache key for rapid lookup of the corresponding NameKeyHash.

Substitute paragraph [0052] with the following:

Incorporation of the random session key in NameHash formation renders it difficult for an attacker of the authentication process to find invalid names that hash to the same value as a valid client name even when the attacker has source code access to the authentication server 105 or has knowledge of confidential aspects of the system 100. The NameHash is employed as a ~~cachekey~~ cache key for storage of client-specific data in the primary cache memory 155, and a later-generated version of the NameHash may be employed to rapidly retrieve the client-specific data from the first cache memory 155.

Substitute paragraph [0061] at page 20 with the following:

In the block 520, the authentication server 105 uses the name from the authentication request as a ~~cachekey~~ cache key to access first validity threshold data from the primary cache 155, such as a stored copy of the client key. Copies of all client names and keys are all valid principals and are stored in the cache 155, which employs the names as ~~cachekeys~~ cache keys to access client-specific data such as copies of the associated client keys from the cache memory 155.

Substitute paragraph [0062] at page 20 with the following:

The cache 155 is updated frequently to maintain currency of the stored data, as is described below in more detail ~~above~~ with reference to Fig. 2. The cache memory 155 may be constructed as a hash table or as any other data structure that allows a quick lookup process to be performed.

Substitute paragraph [0063] at page 20 with the following:

[0063] When the block 520 has employed the client name as a ~~cachekey~~ cache key to access client-specific data from the cache memory 155, control passes to a query task 525.

Substitute paragraph [0064] at page 20 with the following:

In the query task 525, the process 500 determines ~~when~~ whether the client name corresponds to a valid entry in the cache memory 155. When the query task 525 determines that the client name in the authentication request does not correspond to an entry in the cache memory 155, control passes to [[a]] block 530.

Substitute paragraph [0067] at page 21 with the following:

In the query task 535, the process 500 compares client specific data from the authentication request to the client-specific data from the first cache memory 155 to determine ~~when~~ whether the client specific data from the authentication request meet a first threshold of validity.

Substitute paragraph [0068] at page 21 with the following:

For example, the authentication server 105 locates a stored copy of the client key in the primary cache 155 that corresponds to the client name in the authentication request, using the client name as a ~~cachekey~~ cache key. When the query task 535 determines that ~~such~~ the client key matches the data included in the authentication request, the first threshold of validity is met and control passes to [[a]] block 540 and the authentication request proceeds conventionally. When the query task 535 determines that ~~such do~~ the client key does not match the data included in the authentication request, the first threshold of validity is not met, and control passes to the block 530. In either case, the process 500 then ends.

Substitute paragraph [0073] with the following:

In one embodiment, the block 615 causes the authentication server 105 to compute a NameHash using the client name from the authentication request and a RandomSessionKey formed in the authentication server at startup, and uses the NameHash or a truncated version of the NameHash as a ~~cachekey~~ cache key to access first validity threshold data from the primary cache 155, such as a stored copy of the NameHash. The NameHash is computed as NameHash = First X bytes of HMAC(RandomSessionKey, Name).

Substitute paragraph [0074] with the following:

Forming the primary 155 and secondary 160 caches as hash tables, and using the NameHash as a ~~cachekey~~ cache key for the primary cache 155 and the client name as a ~~cachekey~~ cache key for the secondary cache 160, can increase the speed and throughput of the authentication process 600. In one embodiment, the NameHash for all valid clients is stored in the primary cache memory 155 at all times after entry thereof during the updating process.

Substitute paragraph [0076] at page 24 with the following:

In the query task 620, the process 600 determines ~~when~~ whether the primary cache 155 contains an entry corresponding to data contained in the authentication request. ~~When~~ If the query task 620 determines that the primary cache does not contain such an entry, control passes to a block 630 (discussed later below).

Substitute paragraph [0079] at page 24 with the following:

In one embodiment, the authentication server 105 locates a stored copy of the client name or NameHash in the primary cache 155 in the block 620 that corresponds to the client name in the request. When the query task 625 determines that ~~such~~ the client name matches the data included in the authentication request, the first threshold of validity is met, and when the query task 625 determines that ~~such do~~ the client name does not match the data included in the authentication request, the first threshold of validity is not met. When the query task 625 determines that the credentials offered in the authentication request do not meet the first threshold of validity, control passes to the block 630.

Substitute paragraph [0084] with the following:

In the query task 650, the process 600 compares the client specific data to the data from the second cache memory 160 to determine whether the client specific data meet a second threshold of validity. For example, the name associated with the authentication request is used as a ~~cachekey~~ cache key to look up the associated client key and validity flag in the secondary cache memory 160 (e.g., the name is used as a ~~cachekey~~ cache key to obtain the associated data).

Substitute paragraph [0091] with the following:

In the block 715, the authentication server 105 computes a NameHash from the name in the client request to try to access first validity threshold data from the primary cache 155. In one embodiment, the authentication server computes the NameHash using the random session key and the name from the authentication request and employs the NameHash as a ~~cachekey~~ cache key to obtain associated information, such as a stored NameKeyHash. The NameHash is computed as NameHash = First X bytes of HMAC(RandomSessionKey, Name). Alternatively, the NameKeyHash is employed as a ~~cachekey~~ cache key. When the block 715 has tried to obtain the first validity threshold data from the primary cache 155, control passes to a query task 720

Substitute paragraph [0092] at page 27 with the following:

In the query task 720, the process 700 determines ~~when~~ whether the primary cache 155 contains an entry corresponding to data contained in the authentication request. When the query task 720 determines that the primary cache does not contain such an entry, control passes to a block 730 (discussed later below).

Substitute paragraph [0095] at page 28 with the following:

In one embodiment, the authentication server 105 locates a NameKeyHash in the primary cache 155 in the block 720 that corresponds to the name in the request, using the NameHash (either contained in the request or computed in the authorization server 105) as a ~~cachekey~~ cache key. The NameKeyHash retrieved from the first cache memory 155 is then compared to the NameKeyHash from the authentication request in the query task 725. When the query task 725 determines that ~~such~~ the two NameKeyHash values match, the first threshold of validity is met, and when the query task 725 determines that ~~such~~ the two NameKeyHash values do not match, the first threshold of validity is not met. Alternatively, the NameKeyHash may be used as a ~~cachekey~~ cache key to look up the NameHash and then match the retrieved NameHash to a NameHash included in the authorization request.

Substitute paragraph [0101] with the following:

In the query task 750, the process 700 compares the client specific data to the data from the second cache memory 160 to determine whether the client specific data meet a second threshold of validity. For example, the name associated with the authentication request is used as a ~~cachekey~~ cache key to look up the associated client key and validity flag in the secondary cache memory 160 (e.g., the name is used as a ~~cachekey~~ cache key to obtain the associated data). When the query task 750 determines that the client-specific data do not meet the second threshold of validity, e.g., the name corresponds to an invalid validity flag, control passes to the block 730, described above.

Substitute paragraph [0110] with the following:

In the block 820, the authentication server 105 uses the name to try to access first validity threshold data from the primary cache 155. In one embodiment, the authentication server computes a NameHash using the random session key and the name from the authentication request and employs the NameHash as a ~~cachekey~~ cache key to obtain associated information, such as a stored NameKeyHash or truncated version thereof. The NameHash is computed as NameHash = First X bytes of HMAC(RandomSessionKey, Name). Alternatively, the NameKeyHash is computed and employed as a ~~cachekey~~ cache key. When the block 820 has tried to obtain the first validity threshold data from the primary cache 155, control passes to a query task 825.

Substitute paragraph [0111] at page 32 with the following:

In the query task 825, the process 800 determines ~~when~~ whether the NameHash corresponds to an entry in the primary cache 155 and provides related data such as a stored copy of the NameKeyHash.

Substitute paragraph [0112] with the following:

When the query task 825 determines that no valid entry exists in the primary cache memory 155 for the ~~cachekey~~ cache key from the authorization request, control passes to a block 830.

Substitute paragraph [0115] at page 33 with the following:

In the query task 835, the process 800 compares client specific data from the authentication request to the data from the first cache memory 155 to determine ~~when~~ whether the client specific data meet a first threshold of validity.

Substitute paragraph [0118] at page 33 with the following:

In the block 845, the authentication server 105 uses the client name to try to access second validity threshold data from the secondary cache 160. In one embodiment, the authentication server 105 uses the name from the authentication request as a ~~cachekey~~ cache key to obtain associated information, such as a stored copy of the client key and a validity flag. When the block 845 has tried to obtain the second validity threshold data from the secondary cache 160, control passes to a query task 850.

Substitute paragraph [0116] at page 33 with the following:

In one embodiment, the authentication server 105 uses a NameKeyHash or a truncated version thereof from the primary cache 155 to compute a TimedNameKeyHash using the time supplied in the authentication request. The computed TimedNameKeyHash is then compared to a TimedNameKeyHash contained in the authentication request. ~~When~~ If the query task 835 determines that ~~such match~~ the computed TimedNameKeyHash matches the TimedNameKeyHash contained in the authentication request, the first threshold of validity is met, and when the query task 835 determines that ~~such do not match~~ there is no match, the first threshold of validity is not met.

Substitute paragraph [0121] at page 34 with the following:

In the query task 865, the process 800 determines ~~when~~ whether a database entry corresponding to the client name was found in the database 140 in the block 855. ~~When~~ If the query task 865 determines that no such entry exists in the database 140, control passes to the block 830, and the name key pair from the authentication request is then stored in the second cache memory 160 with the appropriate validity flag. ~~When~~ If the second cache memory 160 is already full, an entry in the second cache memory 160 can be evicted using any of a variety of protocols (e.g., according to a least recently used or pseudo least recently used algorithm). When the query task 865 determines that such an entry exists, control passes to the query task 870.